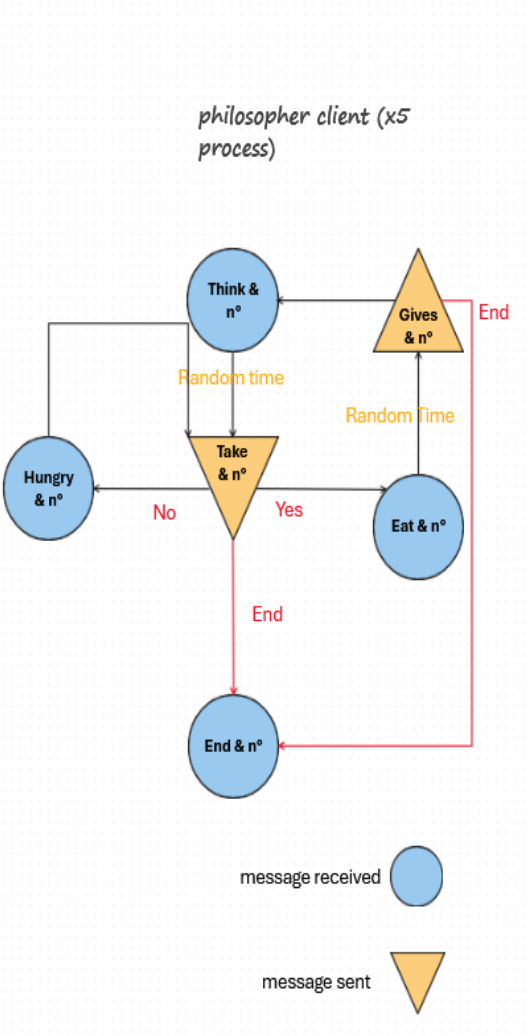
Nicolas
FRERE

# Projet - NS103

Le problème du « **dîner des philosophes** » de Dijkstra permet d'illustrer le phénomène d'interblocage sur des ressources partagées, chacun des philosophes ayant besoin de la fourchette d'un voisin pour manger.

*Benjamin D. Esham / Wikimedia Commons*

Pour représenter ce problème, j'ai effectué deux programmes. Le premier, le client représente les philosophes. Il est composé d'un processus père et de ses fils, qui gèrent chacun indépendamment un socket leur permettant de communiquer en UDP avec l'autre programme, le serveur.



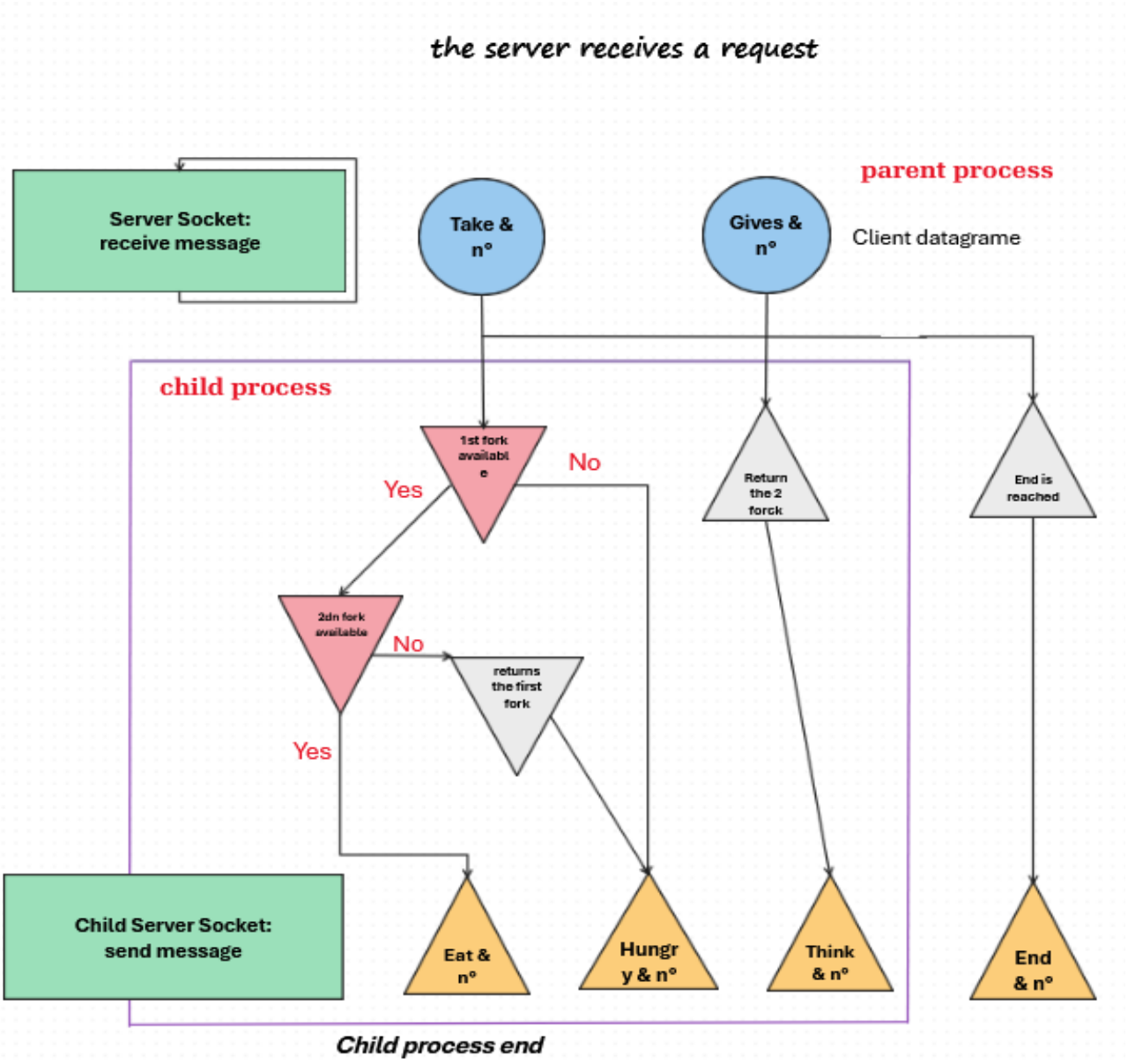

Afin d'empêcher les blocages, le serveur utilise des sémaphores dans une politique de prévention :

Les philosophes ne peuvent garder allouée une fourchette que si les deux sont disponibles (un processus ne démarre que quand les deux ressources lui ont été attribué). Chaque philosophe, après avoir pensé tentera de prendre sa fourchette. Cette opération de type P (demande d'entrée en section critique) sur le sémaphore réalisé avec le flg « IPC_NOWAIT », permet d'avoir un retour instantané sur la disponibilité de la ressource : si elle l'est, l'opération est réalisée, sinon elle ne l'est pas, et « errno » est valorisé à « EAGAIN ». Dans ce cas, on renvoie le philosophe. Affamé, il rentera sa chance un court instant plus tard. Si le philosophe a sa propre fourchette, toujours avec le flg « IPC_NOWAIT », il essayera d'attraper celle d'un plus jeune (n° de fork plus élevé) ou du père pour le plus jeune (un ordre sur l'allocation des ressources est établi). Dans le cas où elle ne serait pas disponible, il reposera sa fourchette et sera dans l'état affamé. Sinon il partira manger pendant un temps indéterminé.

J'ai fait le choix de traiter l'acquisition des fourchettes séparément, mais avec un sembuf de taille 2 cette opération peu être réalisé en une fois.

```
operation[0].sem_num = req.number;
operation[0].sem_flg = IPC_NOWAIT;
operation[0].sem_op = -1;
operation[1].sem_num = (req.number + 1 ) % nbFork;
operation[1].sem_flg = IPC_NOWAIT;
operation[1].sem_op = -1;
semop(sema, operation, 2);
if(errno == EAGAIN  ) {
```

```
/* Processes TAKES requests returns 1
   if two forks are available otherwise zero */
int takeProcessing(int sema , int number)
{
    // With the flag IPC_NOWAIT if the semop operation is not possible
    // the program don't wait, semop don't do the operation
    // and errno == EAGAIN
    operation[0].sem_num = number;
    operation[0].sem_flg = IPC_NOWAIT;
    operation[0].sem_op = -1;
    semop(sema, operation, 1);
    if(errno == EAGAIN) {
        return 0;
    } else {
        operation[0].sem_num = (number + 1 ) % NBFORK;
        operation[0].sem_flg = IPC_NOWAIT;
        operation[0].sem_op = -1;
        semop(sema, operation, 1);
        if(errno == EAGAIN) {
            // If the second fork is not accesible
            // the philosopher must return the first
            givesProcessing(sema, number, 0);
            return 0;
        }
    }
    return 1;
}
```




*En rouge, deux sémaphores manipulés avec une opération de type P*

Pour être plus disponible, le serveur traite en parallèle chaque requête reçue dans un processus fils. Les connexions sont effectuées en UDP, car avec ce protocole, il n'y a pas de buffer ni d'étape de connexions, ce qui facilite la programmation d'une socket communicant avec plusieurs clients. Un segment de mémoire partagé a été créé pour tenir le compte à partir des processus fils, du nombre de fois où les philosophes ont mangé. Quand TOTALFOOD, repas ont été délivré (ici 25), le serveur par le verbe « END » demande aux clients de terminer leur processus.

```c
// Initializes the number of meals taken to 0
mealCompleted[0] = 0;

// The server exchanges until TOTALFOOD is reached
while( mealCompleted[0] < TOTALFOOD) {
    // Receive the datagram
    len = sizeof(cliaddr);
    recvfrom(listenfd, &req, sizeof(req),
            0, (struct sockaddr*)&cliaddr, (unsigned int*)&len);
    // Child process the request
    if ( fork() == 0) {
        responseProcessing(listenfd, req, len, mealCompleted, sema);
    }
}
// TOTALFOOD is reached, the server asks the clients to turn off
closeProcessing(listenfd, req, len, mealCompleted);
```

*Le serveur traite chaque requête depuis un fils dédié.*

```c
/* Deals with the requests of philosophers,
   responds based on demand and availability of forks */
void responseProcessing(int listenfd, struct missive req, int len, int mealCompleted[], int sema)
{
    // Philosopher req.number asks to take forks
    if ((strncmp(req.message, TAKE, 4)) == 0) {
        if (takeProcessing(sema, req.number)) {
            mealCompleted[0]++;
            printf("--Philosophe n° %d mange \n", req.number);
            strcpy(req.message, EAT);
        } else {
            strcpy(req.message, HUNGRY);
        }
    }
    // The philosopher req.number asks to return the forks
    if ((strncmp(req.message, GIVES, 5)) == 0) {
        printf("++Philosophe n° %d rend les fourchettes \n", req.number);
        givesProcessing(sema, req.number, 1);
        strcpy(req.message, THINK);
    }
    // Send the response
    sendto(listenfd, &req, sizeof(req), 0,
        (struct sockaddr*)&cliaddr, len);
    exit(0);
}
```

*Traitement des requêtes reçu coté serveur*

```c
#define PORT 5000
#define TOTALFOOD 25
#define NBFORK 5 // Must correspond to the number of client philosophers

/* Verbs to communicate with the philosophers */
#define TAKE "take"
#define GIVES "gives"
#define THINK "think"
#define HUNGRY "Hungry"
#define EAT "eat"
#define ENDOFMEAL "end"
```

Le port du socket serveur, pour le serveur, le nombre total de fois où les philosophes vont pouvoir manger, et le nombre de philosophes (donc de fourchettes), sont définit dans les entêtes de fichier. PORT, NBFORK de même que les verbes doivent être identiques entre le client et le serveur pour que les programmes fonctionnent.

Pour executer le programme, il faut compiler les deux fichiers :

gcc -Wall -Werror -Wextra udpClient.c -o client

gcc -Wall -Werror -Wextra udpServer.c -o serv

( Les options de compilation -Wall -Werror -Wextra sont optionnel)

Puis il faut les exécuter dans deux terminaux différant sur un système compatible POSIX en commençant par le serveur « ./serv » puis le client « ./client ».

```
gotlub@DESKTOP-LMSLBL3:~/CNAM/NSY103/semaine14$ time ./client
Philosophe n° 3 a mangé 5  fois
Philosophe n° 1 a mangé 5  fois
Philosophe n° 4 a mangé 5  fois
Philosophe n° 0 a mangé 5  fois

real    0m0.014s
user    0m0.001s
sys     0m0.000s
Philosophe n° 2 a mangé 5  fois
```

```
gotlub@DESKTOP-LMSLBL3:~/CNAM/NSY103/semaine14$ ./serv
--Philosophe n° 1 mange
--Philosophe n° 3 mange
++Philosophe n° 3 rend les fourchettes
++Philosophe n° 1 rend les fourchettes
--Philosophe n° 4 mange
--Philosophe n° 2 mange
++Philosophe n° 4 rend les fourchettes
++Philosophe n° 2 rend les fourchettes
--Philosophe n° 3 mange
--Philosophe n° 1 mange
++Philosophe n° 3 rend les fourchettes
++Philosophe n° 1 rend les fourchettes
--Philosophe n° 4 mange
--Philosophe n° 2 mange
++Philosophe n° 4 rend les fourchettes
++Philosophe n° 2 rend les fourchettes
--Philosophe n° 0 mange
--Philosophe n° 2 mange
++Philosophe n° 0 rend les fourchettes
++Philosophe n° 2 rend les fourchettes
--Philosophe n° 4 mange
--Philosophe n° 2 mange
++Philosophe n° 4 rend les fourchettes
++Philosophe n° 2 rend les fourchettes
--Philosophe n° 0 mange
--Philosophe n° 3 mange
```

```
++Philosophe n° 0 rend les fourchettes
++Philosophe n° 3 rend les fourchettes
--Philosophe n° 2 mange
--Philosophe n° 0 mange
++Philosophe n° 2 rend les fourchettes
++Philosophe n° 0 rend les fourchettes
--Philosophe n° 1 mange
--Philosophe n° 3 mange
++Philosophe n° 1 rend les fourchettes
++Philosophe n° 3 rend les fourchettes
--Philosophe n° 2 mange
--Philosophe n° 0 mange
++Philosophe n° 2 rend les fourchettes
++Philosophe n° 0 rend les fourchettes
--Philosophe n° 4 mange
--Philosophe n° 1 mange
++Philosophe n° 4 rend les fourchettes
++Philosophe n° 1 rend les fourchettes
--Philosophe n° 2 mange
--Philosophe n° 4 mange
++Philosophe n° 2 rend les fourchettes
++Philosophe n° 4 rend les fourchettes
--Philosophe n° 2 mange
gotlub@DESKTOP-LMSLBL3:~/CNAM/NSY103/semaine14$
```